Virtual Circuit

2.7

Generated by Doxygen 1.9.7

Chapter 1

User documentation

1.1 Project descripiton

Simulation of a virtual circuit board. The board contains 4 types of logic gates and 2 ¨gates¨ used for transmitting/storing data. Every gate has 2 inputs (except for NOT and OUTPUT), however there are no restrictions on how many other gates can have any given gate as an input.

Gates:

- AND
- OR
- XOR
- NOT
- INPUT
- OUTPUT

The goal is to simulate a system with 5 inputs that only outputs true when the input combination is exactly 11 (01011)

1.2 Simulation and usage

When the right input file is given, the board sets up the circuit After the circuit has been built the user has 3 choices

1. calc
    - given input values the board calculates the value of all gates and prints the OUTPUT gates value
2. print
    - after calc has been called at least once, prints all data from every gate to a given file
3. exit
    - exits the program

1.2.1 Input file syntax

1. The first line of the file should list the name of the gates and amount that is needed for the circuit, and should always end with the OUPUT gate type
    - ex.: INPUT 2 AND 1 OR 3 OUTPUT 2
2. After the first line each line should have 2 or 3 gates, which will indicated the connections betwwen them
    - ex.: AND 0 OR 1 OR 2
    - meaning the AND gate with ID of 0 recives it's inputs from OR ID 1 and OR ID 2
    - INPUT gates don't recive inputs here
3. After the connections have been made the files last line should be: END

There are 6 included test files, test3/4/5 are set up incorrectly for testing, but test1/2 and 5-11.txt are usable

1.2.2 Error handling

The program can detect:

- if the input file doesn't exist
- if the input file has a gate type which doesn't exist
- if the user attepts to connect a gate which doesn't exist
- if not all gates are connected (floating gate)
- if calc is called with too few or too many input values

if any of these errors occure the program won't exit, instead it will display the error and ask for new inputs

1.2.3 Compiling

A makefile is included, it can be run 2 ways either use or test

- use: will run as described
- test: will run a test enviroment with gtest (for jporta)

Chapter 2

Hierarchical Index

2.1 Class Hierarchy

This inheritance list is sorted roughly, but not completely, alphabetically:

Chapter 3

Class Index

3.1 Class List

Here are the classes, structs, unions and interfaces with brief descriptions:

Chapter 4

File Index

4.1 File List

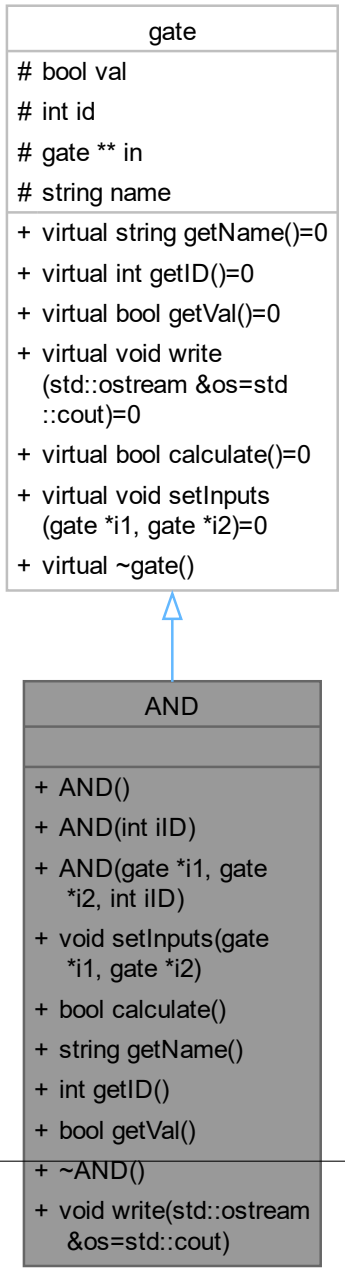
Here is a list of all documented files with brief descriptions:

Chapter 5

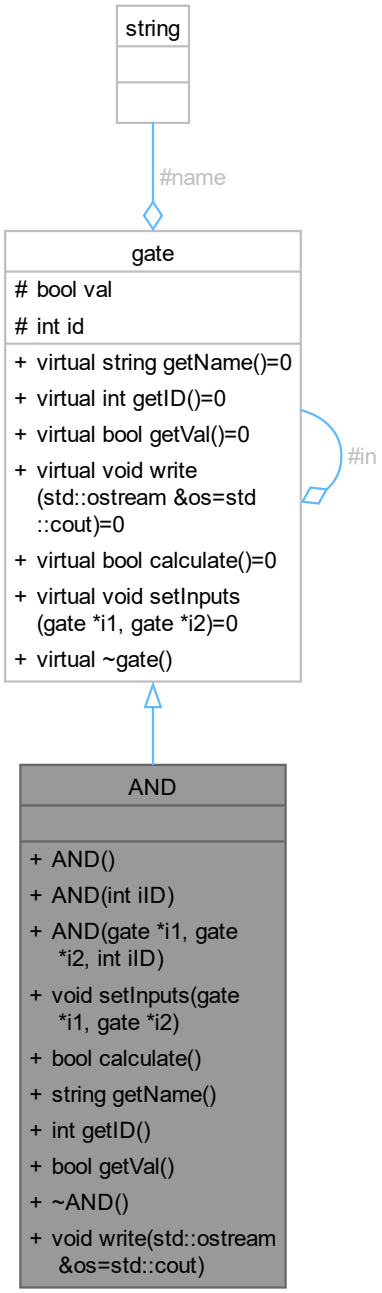
Class Documentation

5.1 AND Class Reference

Inheritance diagram for AND:



Collaboration diagram for AND:



Public Member Functions

- void setInputs (gate *i1, gate *i2)

  if the default constructor is used, can be called to set the input gates
- bool calculate ()

  Calculates value of gate.
- string getName ()

retruns name of gate, used for printing

- int getID ()

  returns id of gate, used for printing
- bool getVal ()

  returns the value of the gate, used for printing
- void write (std::ostream &os=std::cout)

  writes to a given strem: name, id, value

- virtual string getName ()=0

  retruns name of gate, used for printing
- virtual int getID ()=0

  returns id of gate, used for printing
- virtual bool getVal ()=0

  returns the value of the gate, used for printing
- virtual void write (std::ostream &os=std::cout)=0

  writes to a given strem: name, id, value
- virtual bool calculate ()=0

  calculates based on gate type
- virtual void setInputs (gate *i1, gate *i2)=0

  if the default constructor is used, can be called to set the input gates

5.1.1 Member Function Documentation

5.1.1.1 calculate()

```
bool AND::calculate ( )  [virtual]
```

Calculates value of gate.

Calls the calculate() of the 2 input gates and multiplies them together

Returns

true

false

Implements gate.

5.1.1.2 getID()

```
int AND::getID ( )  [virtual]
```

returns id of gate, used for printing

Returns

int

Implements gate.

5.1.1.3 getName()

```
string AND::getName ( )  [virtual]
```

retruns name of gate, used for printing

Returns

string

Implements gate.

5.1.1.4 getVal()

```
bool AND::getVal ( )  [virtual]
```

returns the value of the gate, used for printing

Returns

true

false

Implements gate.

5.1.1.5 setInputs()

```
void AND::setInputs (
            gate * i1,
            gate * i2 )  [virtual]
```

if the default constructor is used, can be called to set the input gates

Parameters

i1	
i2	

Implements gate.

5.1.1.6 write()

```
void AND::write (
            std::ostream & os = std::cout )  [virtual]
```

writes to a given strem: name, id, value

Parameters

os	

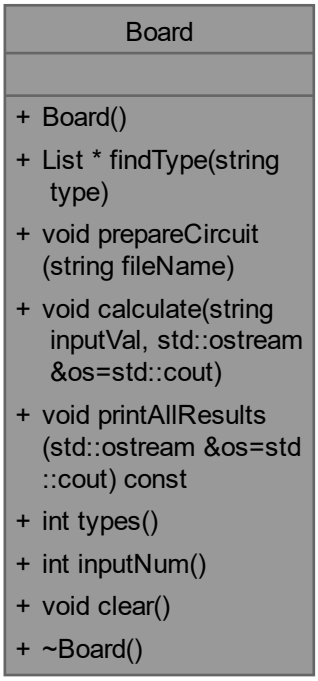
Implements gate.

The documentation for this class was generated from the following files:

- AND.h
- AND.cpp

5.2 Board Class Reference

Collaboration diagram for Board:



Public Member Functions

- List * findType (string type)

  Searches the Lists for a type of gate, returns the List where it found it.
- void prepareCircuit (string fileName)

  Sets up the circuit from a txt file.
- void calculate (string inputVal, std::ostream &os=std::cout)

Calculates the output of the circuit with the given input values.

- void printAllResults (std::ostream &os=std::cout) const

  Outputs all of the gates name, id, value to a given txt file.

- int types ()

  number of different types of gates

- int inputNum ()

  number of input needed (numebr of INPUT-s)

- void clear ()

  resets the board, deletes the lists

5.2.1 Member Function Documentation

5.2.1.1 calculate()

```
void Board::calculate (
            string inputVal,
            std::ostream & os = std::cout )
```

Calculates the output of the circuit with the given input values.

1. finds all of the INPUT-s and sets their signal to the given input
2. finds all the OUTPUT-s and calls calculate() on every instance
3. prints out the value of the OUTPUT-s

Parameters

inputVal	

5.2.1.2 clear()

```
void Board::clear ( )
```

resets the board, deletes the lists

5.2.1.3 findType()

```
List * Board::findType (
            string type )
```

Searches the Lists for a type of gate, returns the List where it found it.

Parameters

type	

Returns

List∗

5.2.1.4 inputNum()

```
int Board::inputNum ( )
```

number of input needed (numebr of INPUT-s)

Returns

int

5.2.1.5 prepareCircuit()

```
void Board::prepareCircuit (
            string fileName )
```

Sets up the circuit from a txt file.

1. creating the gates that are needed for the circuit
2. making the connections between the gates

Parameters

fileName	

5.2.1.6 printAllResults()

```
void Board::printAllResults (
            std::ostream & os = std::cout ) const
```

Outputs all of the gates name, id, value to a given txt file.

Parameters

fileName	

5.2.1.7 types()

```
int Board::types ( )
```

number of different types of gates

Returns

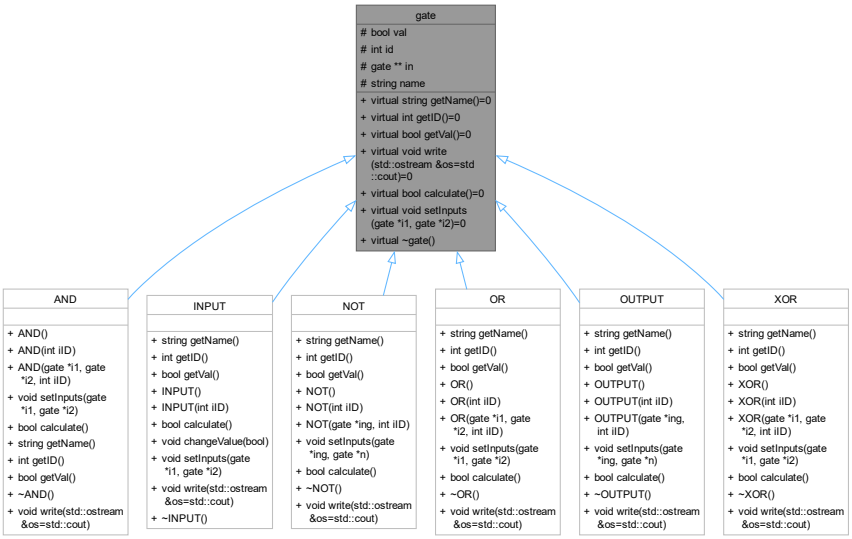
int

The documentation for this class was generated from the following files:

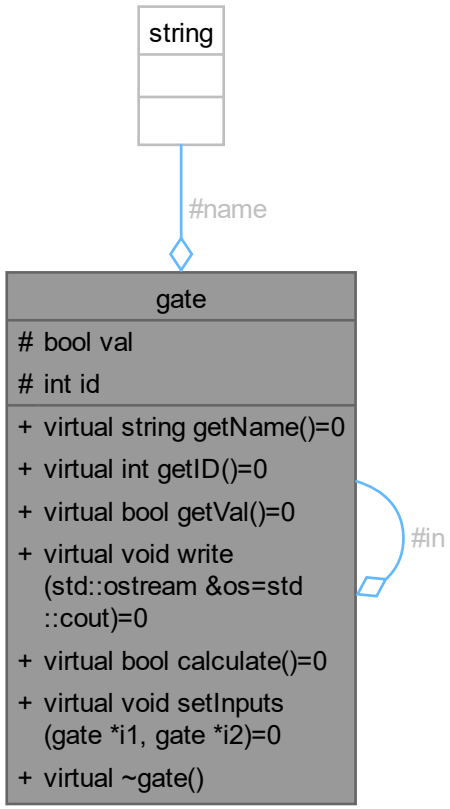
- board.h
- board.cpp

5.3 gate Class Reference

Inheritance diagram for gate:



Collaboration diagram for gate:



Public Member Functions

- virtual string getName ()=0
  retruns name of gate, used for printing
- virtual int getID ()=0
  returns id of gate, used for printing
- virtual bool getVal ()=0
  returns the value of the gate, used for printing
- virtual void write (std::ostream &os=std::cout)=0
  writes to a given strem: name, id, value
- virtual bool calculate ()=0
  calculates based on gate type
- virtual void setInputs (gate *i1, gate *i2)=0
  if the default constructor is used, can be called to set the input gates

5.3.1 Member Function Documentation

5.3.1.1 calculate()

```
virtual bool gate::calculate ( )  [pure virtual]
```

calculates based on gate type

Returns

true
false

Implemented in AND, INPUT, NOT, OR, OUTPUT, and XOR.

5.3.1.2 getID()

```
virtual int gate::getID ( )  [pure virtual]
```

returns id of gate, used for printing

Returns

int

Implemented in AND, INPUT, NOT, OR, OUTPUT, and XOR.

5.3.1.3 getName()

```
virtual string gate::getName ( )  [pure virtual]
```

retruns name of gate, used for printing

Returns

string

Implemented in AND, INPUT, NOT, OR, OUTPUT, and XOR.

5.3.1.4 getVal()

```
virtual bool gate::getVal ( )  [pure virtual]
```

returns the value of the gate, used for printing

Returns

true
false

Implemented in AND, INPUT, NOT, OR, OUTPUT, and XOR.

5.3.1.5 setInputs()

```
virtual void gate::setInputs (
            gate * i1,
            gate * i2 )  [pure virtual]
```

if the default constructor is used, can be called to set the input gates

Parameters

i1	
i2	

Implemented in AND, INPUT, OR, XOR, NOT, and OUTPUT.

5.3.1.6 write()

```
virtual void gate::write (
            std::ostream & os = std::cout )  [pure virtual]
```

writes to a given strem: name, id, value

Parameters

os	

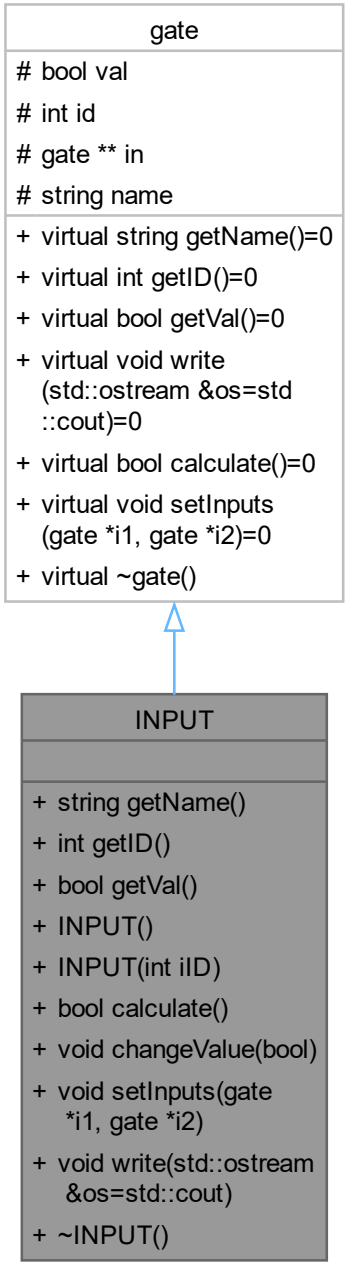
Implemented in AND, INPUT, NOT, OR, OUTPUT, and XOR.

The documentation for this class was generated from the following file:

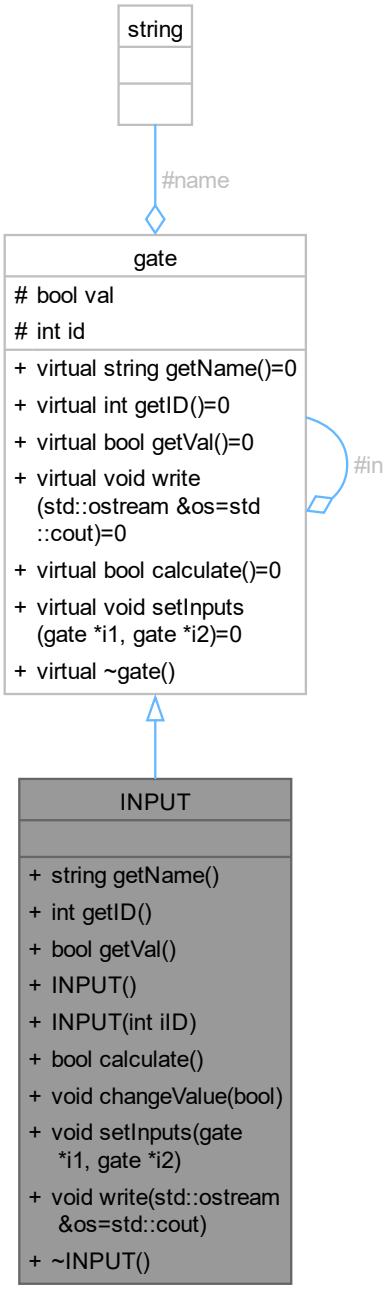
- gate.h

5.4 INPUT Class Reference

Inheritance diagram for INPUT:



Collaboration diagram for INPUT:



Public Member Functions

- string getName ()

  retruns name of gate, used for printing
- int getID ()

  returns id of gate, used for printing
- bool getVal ()

returns the value of the gate, used for printing

- bool calculate ()

  no calculations done just sand signal forward
- void changeValue (bool)

  extra function in INPUT so the value can be changed without needing to rebuild the circuit
- void setInputs (gate *i1, gate *i2)

  empty function since this doesnt have any inputs
- void write (std::ostream &os=std::cout)

  writes to a given strem: name, id, value
- ~INPUT ()

  only empty destructor since no input gates

- virtual string getName ()=0

  retruns name of gate, used for printing
- virtual int getID ()=0

  returns id of gate, used for printing
- virtual bool getVal ()=0

  returns the value of the gate, used for printing
- virtual void write (std::ostream &os=std::cout)=0

  writes to a given strem: name, id, value
- virtual bool calculate ()=0

  calculates based on gate type
- virtual void setInputs (gate *i1, gate *i2)=0

  if the default constructor is used, can be called to set the input gates

5.4.1 Constructor & Destructor Documentation

5.4.1.1 ~INPUT()

```
INPUT::~INPUT ( )  [inline]
```

only empty destructor since no input gates

5.4.2 Member Function Documentation

5.4.2.1 calculate()

```
bool INPUT::calculate ( )  [virtual]
```

no calculations done just sand signal forward

Returns

true

false

Implements gate.

5.4.2.2 changeValue()

```
void INPUT::changeValue (
            bool in )
```

extra function in INPUT so the value can be changed without needing to rebuild the circuit

Parameters

in	

5.4.2.3 getID()

```
int INPUT::getID ( )  [virtual]
```

returns id of gate, used for printing

Returns

int

Implements gate.

5.4.2.4 getName()

```
string INPUT::getName ( )  [virtual]
```

retruns name of gate, used for printing

Returns

string

Implements gate.

5.4.2.5 getVal()

```
bool INPUT::getVal ( )  [virtual]
```

returns the value of the gate, used for printing

Returns

true

false

Implements gate.

5.4.2.6 setInputs()

```
void INPUT::setInputs (
            gate * i1,
            gate * i2 )  [inline], [virtual]
```

empty function since this doesnt have any inputs

Parameters

i1	
i2	

Implements gate.

5.4.2.7 write()

```
void INPUT::write (
            std::ostream & os = std::cout )  [virtual]
```

writes to a given strem: name, id, value

Parameters

os	

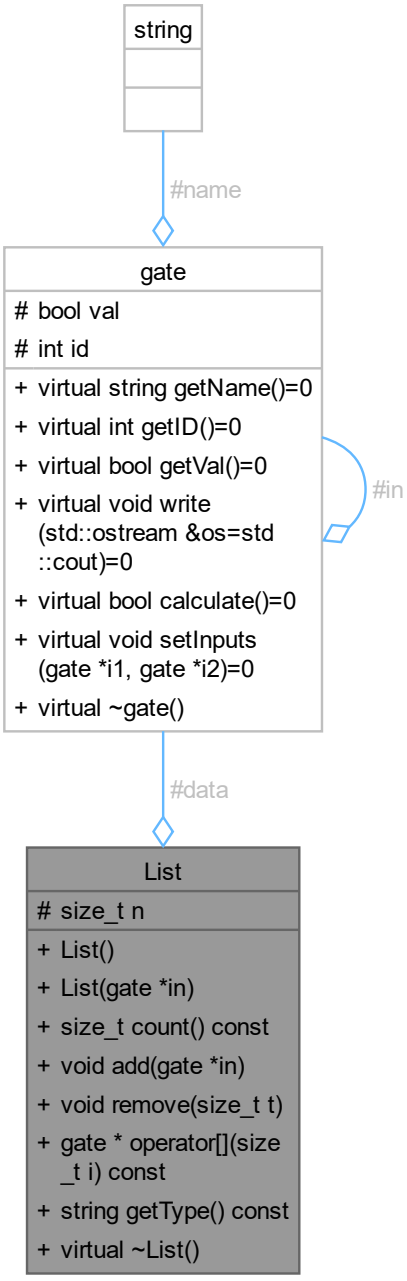
Implements gate.

The documentation for this class was generated from the following files:

- INPUT.h
- INPUT.cpp

5.5 List Class Reference

Collaboration diagram for List:



Public Member Functions

- size_t count () const

  returns the current number of gates in the list

- void add (gate ∗in)

  adds a gate to the end ofthe list like push_back
- void remove (size_t t)

  removes the element at the given index
- gate ∗ operator[ ] (size_t i) const

  index operator, returns data-s element at the given index
- string getType () const

  get the type of gate that is stored in the list
  can decide if the stored gates are heterogenic or homogenic

5.5.1 Member Function Documentation

5.5.1.1 add()

```
void List::add (
            gate * in )
```

adds a gate to the end ofthe list like push_back

Parameters

in	

5.5.1.2 count()

```
size_t List::count ( ) const
```

returns the current number of gates in the list

Returns

size_t

5.5.1.3 getType()

```
string List::getType ( ) const
```

get the type of gate that is stored in the list
can decide if the stored gates are heterogenic or homogenic

Returns

string

5.5.1.4 operator[]()

```
gate * List::operator[] (
            size_t i ) const
```

index operator, returns data-s element at the given index

Parameters

i	

Returns

gate*

5.5.1.5 remove()

```
void List::remove (
            size_t t )
```

removes the element at the given index

Parameters

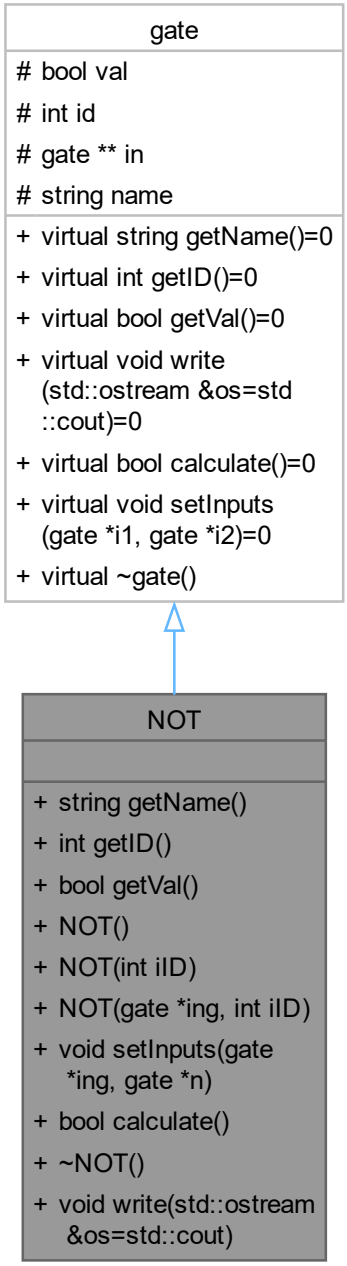
t	

The documentation for this class was generated from the following files:

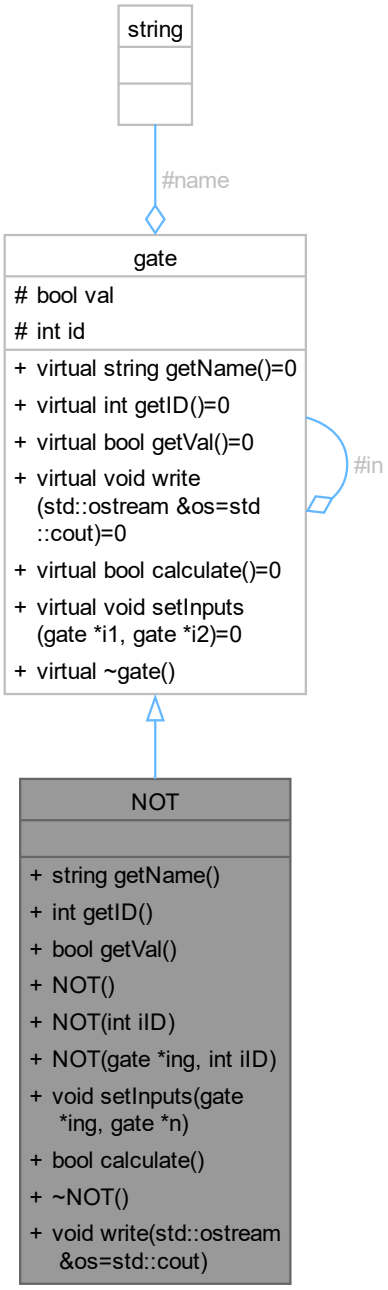
- list.h
- list.cpp

5.6 NOT Class Reference

Inheritance diagram for NOT:



Collaboration diagram for NOT:



Public Member Functions

- string getName ()

  retruns name of gate, used for printing
- int getID ()

  returns id of gate, used for printing
- bool getVal ()

returns the value of the gate, used for printing

- void setInputs (gate *ing, gate *n)

  if the default constructor is used, can be called to set the input gates
- bool calculate ()

  flips incloming signal
- void write (std::ostream &os=std::cout)

  writes to a given strem: name, id, value

- virtual string getName ()=0

  retruns name of gate, used for printing
- virtual int getID ()=0

  returns id of gate, used for printing
- virtual bool getVal ()=0

  returns the value of the gate, used for printing
- virtual void write (std::ostream &os=std::cout)=0

  writes to a given strem: name, id, value
- virtual bool calculate ()=0

  calculates based on gate type
- virtual void setInputs (gate *i1, gate *i2)=0

  if the default constructor is used, can be called to set the input gates

5.6.1 Member Function Documentation

5.6.1.1 calculate()

```
bool NOT::calculate ( )  [virtual]
```

flips incloming signal

Returns

true

false

Implements gate.

5.6.1.2 getID()

```
int NOT::getID ( )  [virtual]
```

returns id of gate, used for printing

Returns

int

Implements gate.

5.6.1.3 getName()

```
string NOT::getName ( )  [virtual]
```

retruns name of gate, used for printing

Returns

string

Implements gate.

5.6.1.4 getVal()

```
bool NOT::getVal ( )  [virtual]
```

returns the value of the gate, used for printing

Returns

true

false

Implements gate.

5.6.1.5 setInputs()

```
void NOT::setInputs (
            gate * i1,
            gate * i2 )  [virtual]
```

if the default constructor is used, can be called to set the input gates

Parameters

i1	
i2	

Implements gate.

5.6.1.6 write()

```
void NOT::write (
            std::ostream & os = std::cout )  [virtual]
```

writes to a given strem: name, id, value

Parameters

os	

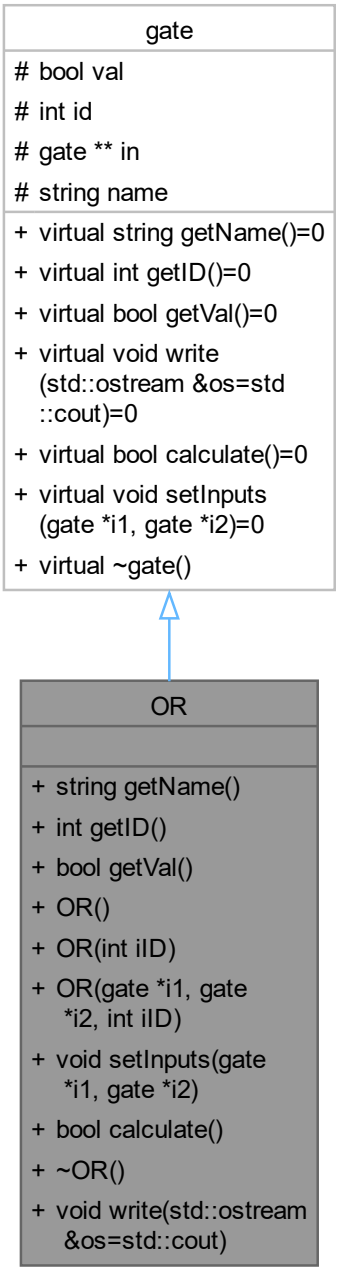
Implements gate.

The documentation for this class was generated from the following files:

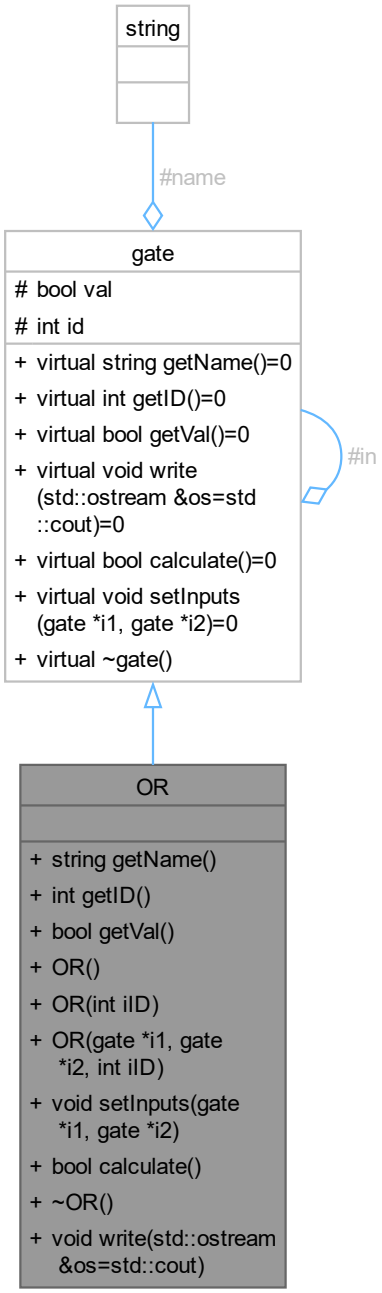
- NOT.h
- NOT.cpp

5.7 OR Class Reference

Inheritance diagram for OR:



Collaboration diagram for OR:



Public Member Functions

- string getName ()

  retruns name of gate, used for printing
- int getID ()

  returns id of gate, used for printing
- bool getVal ()

returns the value of the gate, used for printing

- void setInputs (gate *i1, gate *i2)

  if the default constructor is used, can be called to set the input gates
- bool calculate ()

  Calculates value of gate.
- void write (std::ostream &os=std::cout)

  writes to a given strem: name, id, value

- virtual string getName ()=0

  retruns name of gate, used for printing
- virtual int getID ()=0

  returns id of gate, used for printing
- virtual bool getVal ()=0

  returns the value of the gate, used for printing
- virtual void write (std::ostream &os=std::cout)=0

  writes to a given strem: name, id, value
- virtual bool calculate ()=0

  calculates based on gate type
- virtual void setInputs (gate *i1, gate *i2)=0

  if the default constructor is used, can be called to set the input gates

5.7.1 Member Function Documentation

5.7.1.1 calculate()

```
bool OR::calculate ( )  [virtual]
```

Calculates value of gate.

adds the incoming 2 signals together

Returns

true

false

Implements gate.

5.7.1.2 getID()

```
int OR::getID ( )  [virtual]
```

returns id of gate, used for printing

Returns

int

Implements gate.

5.7.1.3 getName()

```
string OR::getName ( )  [virtual]
```

retruns name of gate, used for printing

Returns

string

Implements gate.

5.7.1.4 getVal()

```
bool OR::getVal ( )  [virtual]
```

returns the value of the gate, used for printing

Returns

true
false

Implements gate.

5.7.1.5 setInputs()

```
void OR::setInputs (
            gate * i1,
            gate * i2 )  [virtual]
```

if the default constructor is used, can be called to set the input gates

Parameters

i1	
i2	

Implements gate.

5.7.1.6 write()

```
void OR::write (
            std::ostream & os = std::cout )  [virtual]
```

writes to a given strem: name, id, value

Parameters

os	

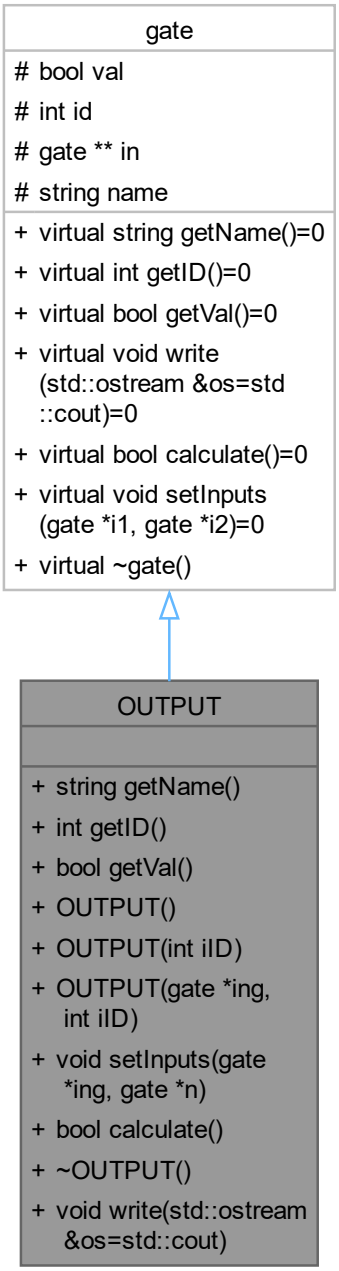
Implements gate.

The documentation for this class was generated from the following files:

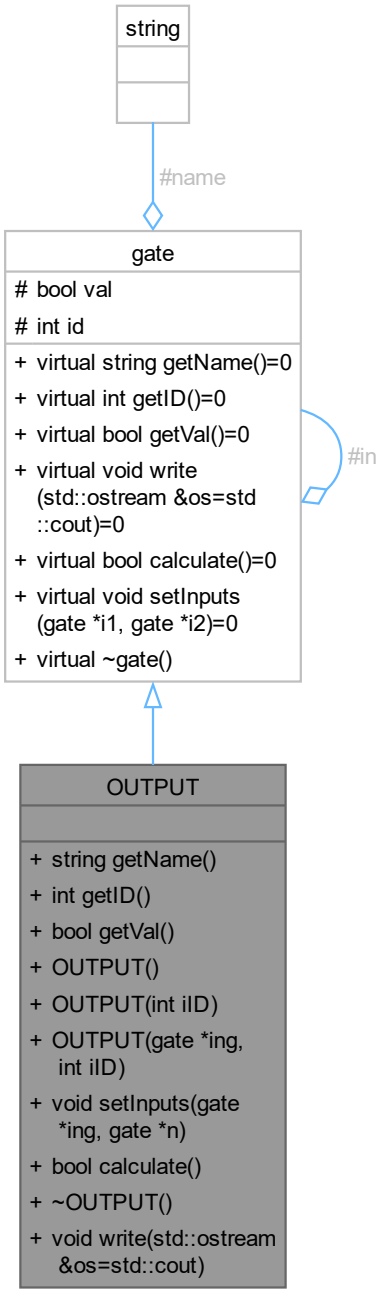
- OR.h
- OR.cpp

5.8 OUTPUT Class Reference

Inheritance diagram for OUTPUT:



Collaboration diagram for OUTPUT:



Public Member Functions

- string getName ()

  retruns name of gate, used for printing
- int getID ()

  returns id of gate, used for printing
- bool getVal ()

returns the value of the gate, used for printing

- void setInputs (gate *ing, gate *n)

  if the default constructor is used, can be called to set the input gates

- bool calculate ()

  Calculates value of gate.

- void write (std::ostream &os=std::cout)

  writes to a given strem: name, id, value

- virtual string getName ()=0

  retruns name of gate, used for printing

- virtual int getID ()=0

  returns id of gate, used for printing

- virtual bool getVal ()=0

  returns the value of the gate, used for printing

- virtual void write (std::ostream &os=std::cout)=0

  writes to a given strem: name, id, value

- virtual bool calculate ()=0

  calculates based on gate type

- virtual void setInputs (gate *i1, gate *i2)=0

  if the default constructor is used, can be called to set the input gates

5.8.1 Member Function Documentation

5.8.1.1 calculate()

```
bool OUTPUT::calculate ( )  [virtual]
```

Calculates value of gate.

doesn't change the incoming signal just stores it

Returns

true

false

Implements gate.

5.8.1.2 getID()

```
int OUTPUT::getID ( )  [virtual]
```

returns id of gate, used for printing

Returns

int

Implements gate.

5.8.1.3 getName()

```
string OUTPUT::getName ( )  [virtual]
```

retruns name of gate, used for printing

Returns

string

Implements gate.

5.8.1.4 getVal()

```
bool OUTPUT::getVal ( )  [virtual]
```

returns the value of the gate, used for printing

Returns

true

false

Implements gate.

5.8.1.5 setInputs()

```
void OUTPUT::setInputs (
            gate * i1,
            gate * i2 )  [virtual]
```

if the default constructor is used, can be called to set the input gates

Parameters

i1	
i2	

Implements gate.

5.8.1.6 write()

```
void OUTPUT::write (
            std::ostream & os = std::cout )  [virtual]
```

writes to a given strem: name, id, value

Parameters

os	

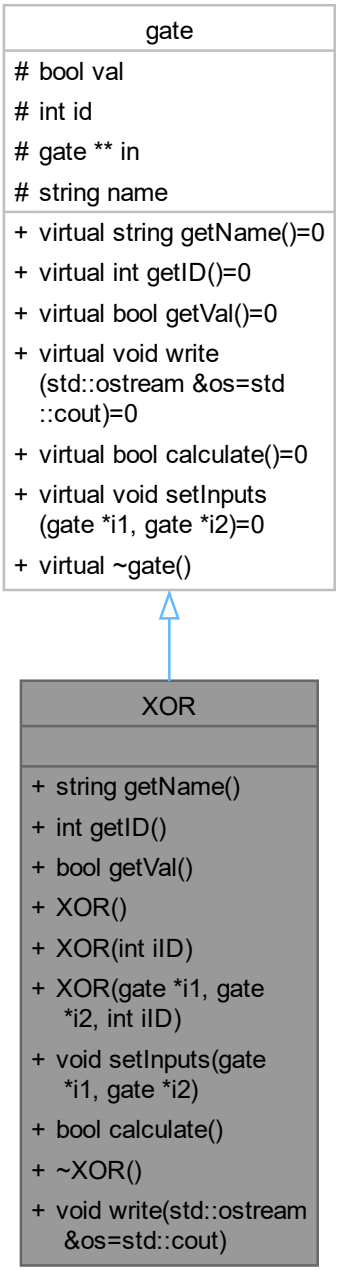
Implements gate.

The documentation for this class was generated from the following files:

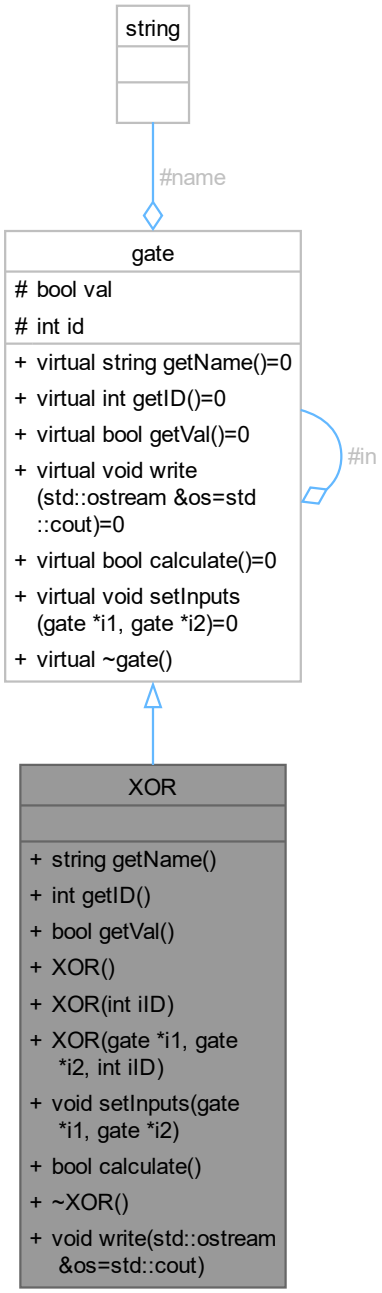
- OUTPUT.h
- OUTPUT.cpp

5.9 XOR Class Reference

Inheritance diagram for XOR:



Collaboration diagram for XOR:



Public Member Functions

- string getName ()

  retruns name of gate, used for printing
- int getID ()

  returns id of gate, used for printing
- bool getVal ()

returns the value of the gate, used for printing

- void setInputs (gate *i1, gate *i2)

  if the default constructor is used, can be called to set the input gates
- bool calculate ()

  Calculates value of gate.
- void write (std::ostream &os=std::cout)

  writes to a given strem: name, id, value

- virtual string getName ()=0

  retruns name of gate, used for printing
- virtual int getID ()=0

  returns id of gate, used for printing
- virtual bool getVal ()=0

  returns the value of the gate, used for printing
- virtual void write (std::ostream &os=std::cout)=0

  writes to a given strem: name, id, value
- virtual bool calculate ()=0

  calculates based on gate type
- virtual void setInputs (gate *i1, gate *i2)=0

  if the default constructor is used, can be called to set the input gates

5.9.1 Member Function Documentation

5.9.1.1 calculate()

```
bool XOR::calculate ( )  [virtual]
```

Calculates value of gate.

calculates the exclusive-or of the two incoming signals

Returns

true

false

Implements gate.

5.9.1.2 getID()

```
int XOR::getID ( )  [virtual]
```

returns id of gate, used for printing

Returns

int

Implements gate.

5.9.1.3 getName()

```
string XOR::getName ( )  [virtual]
```

retruns name of gate, used for printing

Returns

string

Implements gate.

5.9.1.4 getVal()

```
bool XOR::getVal ( )  [virtual]
```

returns the value of the gate, used for printing

Returns

true

false

Implements gate.

5.9.1.5 setInputs()

```
void XOR::setInputs (
            gate * i1,
            gate * i2 )  [virtual]
```

if the default constructor is used, can be called to set the input gates

Parameters

i1	
i2	

Implements gate.

5.9.1.6 write()

```
void XOR::write (
            std::ostream & os = std::cout )  [virtual]
```

writes to a given strem: name, id, value

Parameters

os	

Implements gate.

The documentation for this class was generated from the following files:

- XOR.h
- XOR.cpp

Chapter 6

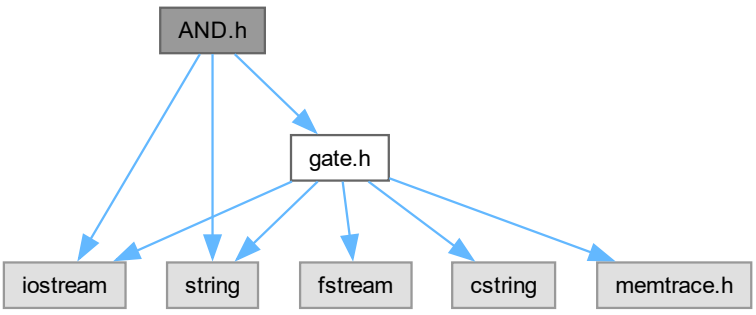
File Documentation

6.1 AND.h File Reference

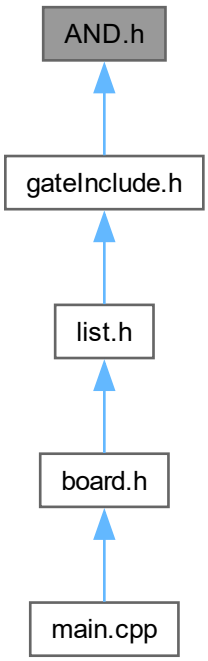
Implementation of the AND gate, 2 inputs, mutiplies incoming signals.

```
#include <iostream>
#include <string>
#include ¨gate.h¨
```

Include dependency graph for AND.h:



This graph shows which files directly or indirectly include this file:



Classes

- class AND

6.1.1 Detailed Description

Implementation of the AND gate, 2 inputs, mutiplies incoming signals.

6.2 AND.h

Go to the documentation of this file.

```
00001
00006 #ifndef ANDG
00007 #define ANDG
00008
00009 #include <iostream>
00010 #include <string>
00011 #include "gate.h"
00012
00013 using std::string;
00014 using std::cout;
00015 using std::endl;
00016
00017 class AND : public gate
00018 {
00019     string name = "AND";
00020 public:
```

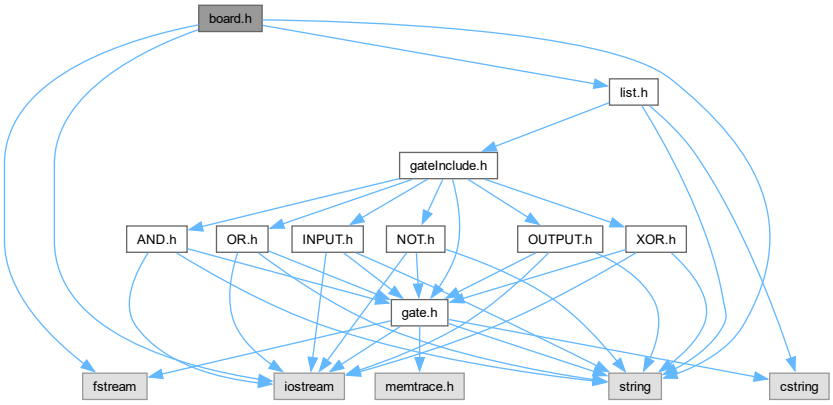
```
00021     AND();
00022     AND(int iID);
00023     AND(gate *i1, gate *i2, int iID);
00024     void setInputs(gate *i1, gate *i2);
00025
00032     bool calculate();
00033
00034     string getName();
00035     int getID();
00036     bool getVal();
00037
00038     ~AND();
00039     void write(std::ostream& os = std::cout);
00040 };
00041
00042
00043 #endif
```

6.3 board.h File Reference

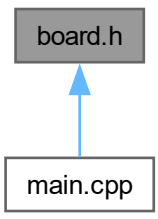
contains a dinamic list of gates, responsible for all the alculations and file management

```
#include <iostream>
#include <fstream>
#include <string>
#include ¨list.h¨
```

Include dependency graph for board.h:



This graph shows which files directly or indirectly include this file:



Classes

- class Board

6.3.1 Detailed Description

contains a dinamic list of gates, responsible for all the alculations and file management

6.4 board.h

Go to the documentation of this file.

```
00001
00006 #ifndef BOARD
00007 #define BOARD
00008
00009 #include <iostream>
00010 #include <fstream>
00011 #include <string>
00012 #include "list.h"
00013
00014 using std::string;
00015 using std::cout;
00016 using std::endl;
00017
00018 class Board
00019 {
00020     List* gates;
00021     size_t typeCount = 0;
00022 public:
00023     Board(){}
00024
00031     List* findType(string type);
00032
00040     void prepareCircuit(string fileName);
00041
00050     void calculate(string inputVal, std::ostream& os = std::cout);
00051
00052
00058     void printAllResults(std::ostream& os = std::cout) const;
00059
00065     int types();
00066
00072     int inputNum();
00073
00078     void clear();
00079
00080     ~Board();
00081 };
00082
00083
00084 #endif
```

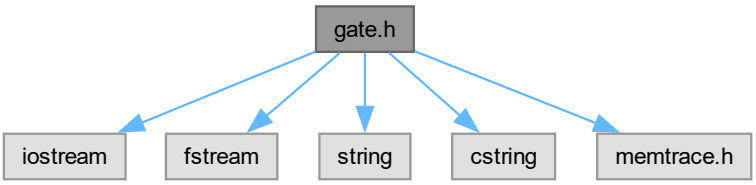
6.5 gate.h File Reference

abstract parent class of every gate

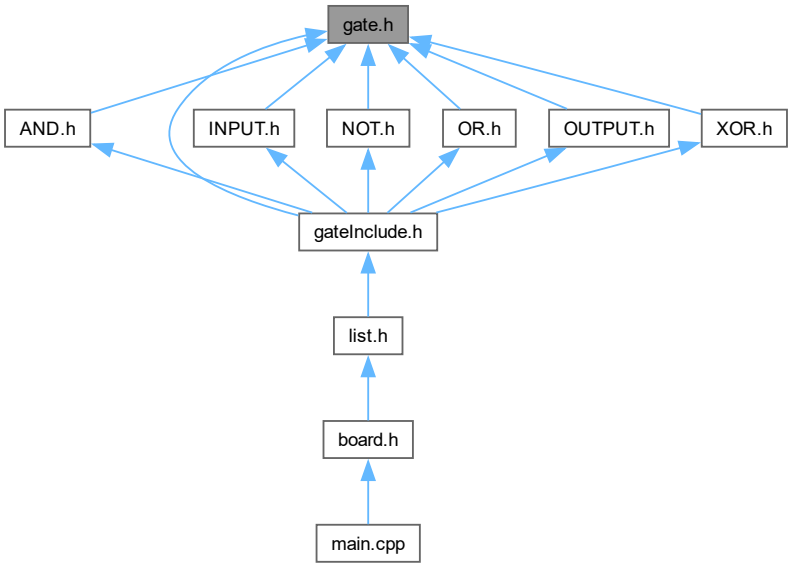
```
#include <iostream>
#include <fstream>
#include <string>
#include <cstring>
```

```
#include ¨memtrace.h¨
```

Include dependency graph for gate.h:



This graph shows which files directly or indirectly include this file:



Classes

- class gate

6.5.1 Detailed Description

abstract parent class of every gate

6.6 gate.h

Go to the documentation of this file.

```
00001
00005 #ifndef GATE
00006 #define GATE
00007
00008 #include <iostream>
00009 #include <fstream>
00010 #include <string>
00011 #include <cstring>
00012 #include "memtrace.h"
00013
00014 using std::string;
00015 using std::cout;
00016 using std::endl;
00017
00018 class gate
00019 {
00020 protected:
00021
00022     bool val = NULL;
00023     int id = 0;
00024     gate **in;
00025     string name = "gate";
00026
00027 public:
00028
00034     virtual string getName() = 0;
00035
00041     virtual int getID() = 0;
00042
00049     virtual bool getVal() = 0;
00050
00056     virtual void write(std::ostream& os = std::cout) = 0;
00057
00064     virtual bool calculate() = 0;
00065
00072     virtual void setInputs(gate *i1, gate *i2) = 0;
00073     virtual ~gate(){};
00074 };
00075
00076
00077 #endif
```

6.7 gateInclude.h

```
00001 #ifndef GATEIN
00002 #define GATEIN
00003
00004 #include "gate.h"
00005 #include "AND.h"
00006 #include "OR.h"
00007 #include "XOR.h"
00008 #include "NOT.h"
00009 #include "INPUT.h"
00010 #include "OUTPUT.h"
00011
00012 #endif
```

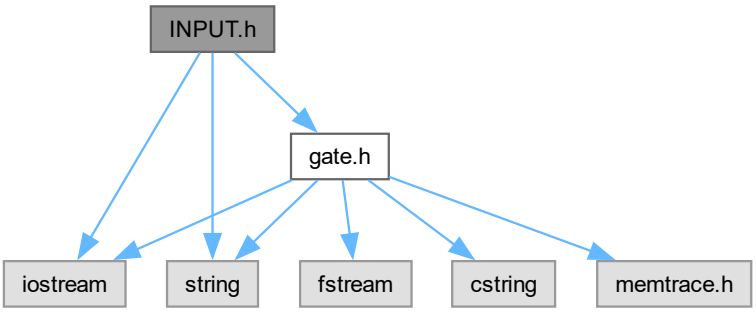
6.8 INPUT.h File Reference

Implementation of the INPUT, first line of ¨gates¨ no inputs just sends a given signal forward.

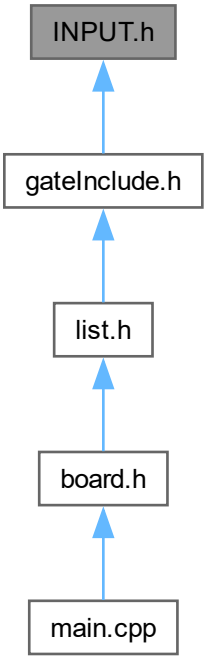
```
#include <iostream>
#include <string>
```

```
#include ¨gate.h¨
```

Include dependency graph for INPUT.h:



This graph shows which files directly or indirectly include this file:



Classes

- class INPUT

6.8.1 Detailed Description

Implementation of the INPUT, first line of ¨gates¨ no inputs just sends a given signal forward.

6.9 INPUT.h

Go to the documentation of this file.

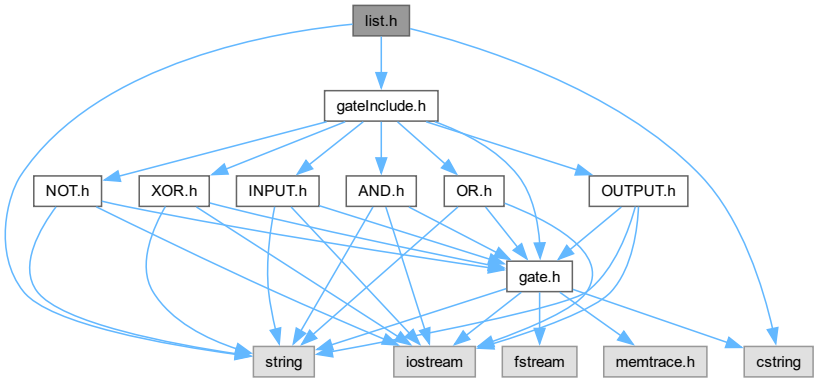
```
00001
00006 #ifndef INPUTG
00007 #define INPUTG
00008
00009 #include <iostream>
00010 #include <string>
00011 #include "gate.h"
00012
00013 using std::string;
00014 using std::cout;
00015 using std::endl;
00016
00017 class INPUT : public gate
00018 {
00019     string name = "INPUT";
00020 public:
00021     string getName();
00022     int getID();
00023     bool getVal();
00024
00025     INPUT(){}
00026     INPUT(int iID);
00027
00034     bool calculate();
00035
00041     void changeValue(bool);
00042
00049     void setInputs(gate *i1, gate *i2){}
00050
00051     void write(std::ostream& os = std::cout);
00052
00057     ~INPUT(){}
00058
00059
00060 };
00061
00062
00063 #endif
```

6.10 list.h File Reference

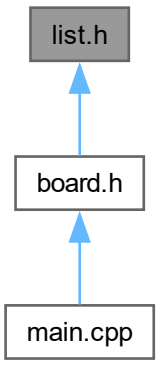
contains a dynamic array of gate pointers

```
#include <string>
#include <cstring>
#include ¨gateInclude.h¨
```

Include dependency graph for list.h:



This graph shows which files directly or indirectly include this file:



Classes

- class List

6.10.1 Detailed Description

contains a dynamic array of gate pointers

6.11 list.h

Go to the documentation of this file.

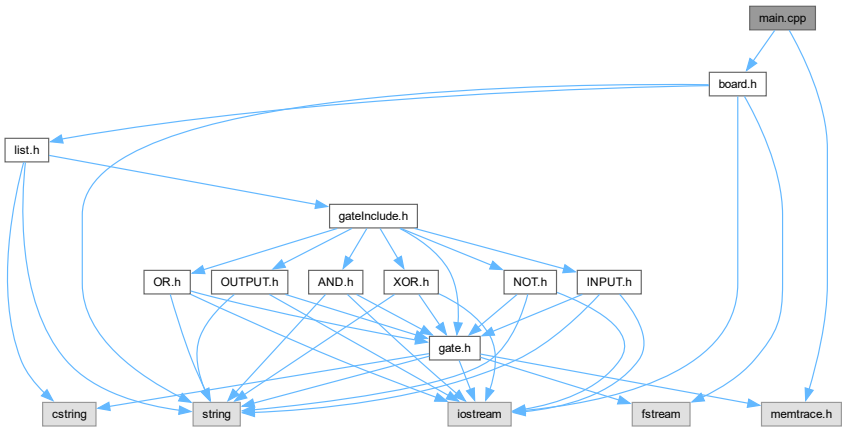
```
00001
00006 #ifndef LIST
00007 #define LIST
00008
00009 #include <string>
00010 #include <cstring>
00011 #include "gateInclude.h"
00012
00013 class List
00014 {
00015 protected:
00016     size_t n = 0;
00017     gate **data = nullptr;
00018
00019 public:
00020
00021     List(){}
00022
00023     List(gate *in);
00024
00030     size_t count() const;
00031
00037     void add(gate *in);
00038
00044     void remove(size_t t);
00045
00052     gate* operator[](size_t i) const;
00053
00060     string getType() const;
00061     virtual ~List();
00062 };
00063
00064 #endif
```

6.12 main.cpp File Reference

includes the main user interface handling

```
#include ¨board.h¨
#include ¨memtrace.h¨
```

Include dependency graph for main.cpp:



6.12.1 Detailed Description

includes the main user interface handling

Author

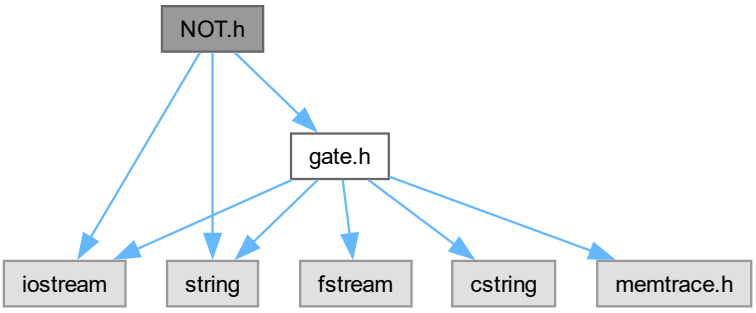
Tóth Milán György
LPDC71

6.13 NOT.h File Reference

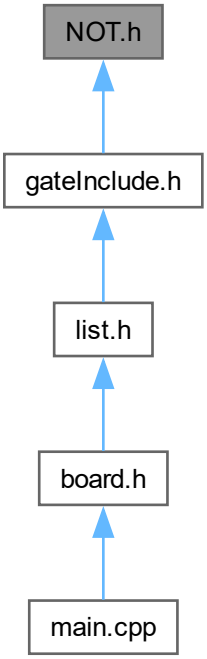
Implementation of the NOT gate, 1 input, flips incoming signal.

```
#include <iostream>
#include <string>
#include ¨gate.h¨
```

Include dependency graph for NOT.h:



This graph shows which files directly or indirectly include this file:



Classes

- class NOT

6.13.1 Detailed Description

Implementation of the NOT gate, 1 input, flips incoming signal.

6.14 NOT.h

Go to the documentation of this file.

```
00001
00006 #ifndef NOTG
00007 #define NOTG
00008
00009 #include <iostream>
00010 #include <string>
00011 #include "gate.h"
00012
00013 using std::string;
00014 using std::cout;
00015 using std::endl;
00016
00017 class NOT : public gate
00018 {
00019     string name = "NOT";
00020 public:
```

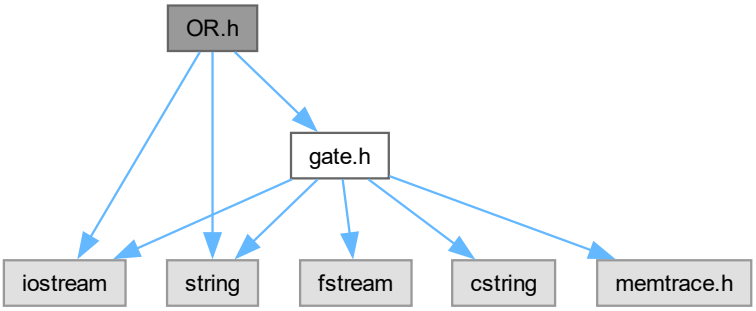
```
00021     string getName();
00022     int getID();
00023     bool getVal();
00024 
00025     NOT();
00026     NOT(int iID);
00027     NOT(gate *ing, int iID);
00028     void setInputs(gate *ing, gate *n);
00029 
00036     bool calculate();
00037     ~NOT();
00038 
00039     void write(std::ostream& os = std::cout);
00040 };
00041 
00042 
00043 #endif
```

6.15 OR.h File Reference

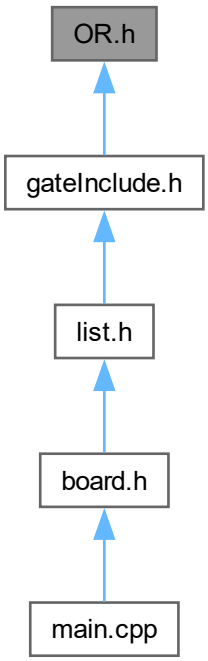
Implementation of the OR gate, 2 inputs, adds incoming singals.

```
#include <iostream>
#include <string>
#include ¨gate.h¨
```

Include dependency graph for OR.h:



This graph shows which files directly or indirectly include this file:



Classes

- class OR

6.15.1 Detailed Description

Implementation of the OR gate, 2 inputs, adds incoming singals.

6.16 OR.h

Go to the documentation of this file.

```
00001
00006 #ifndef ORG
00007 #define ORG
00008
00009 #include <iostream>
00010 #include <string>
00011 #include "gate.h"
00012
00013 using std::string;
00014 using std::cout;
00015 using std::endl;
00016
00017 class OR : public gate
00018 {
00019     string name = "OR";
00020 public:
```

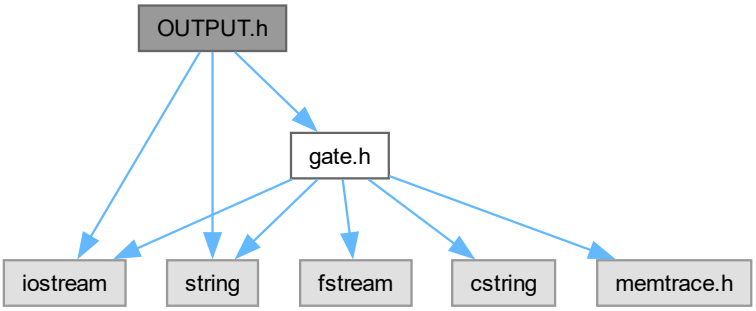
```
00021     string getName();
00022     int getID();
00023     bool getVal();
00024 
00025     OR();
00026     OR(int iID);
00027     OR(gate *i1, gate *i2, int iID);
00028     void setInputs(gate *i1, gate *i2);
00029 
00036     bool calculate();
00037     ~OR();
00038 
00039     void write(std::ostream& os = std::cout);
00040 };
00041 
00042 
00043 #endif
```

6.17 OUTPUT.h File Reference

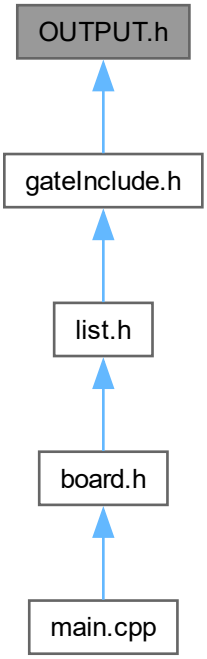
Implementation of the OUTPUT, 1 input, only saves and passes along the signal.

```
#include <iostream>
#include <string>
#include ¨gate.h¨
```

Include dependency graph for OUTPUT.h:



This graph shows which files directly or indirectly include this file:



Classes

- class OUTPUT

6.17.1 Detailed Description

Implementation of the OUTPUT, 1 input, only saves and passes along the signal.

6.18 OUTPUT.h

Go to the documentation of this file.

```
00001
00006 #ifndef OUTPUTG
00007 #define OUTPUTG
00008
00009 #include <iostream>
00010 #include <string>
00011 #include "gate.h"
00012
00013 using std::string;
00014 using std::cout;
00015 using std::endl;
00016
00017 class OUTPUT : public gate
00018 {
00019     string name = "OUTPUT";
00020 public:
```

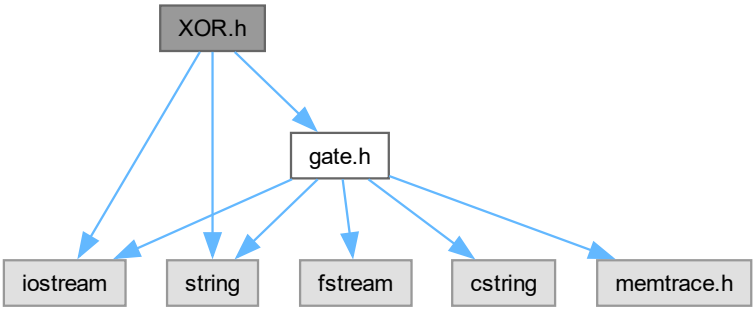
```
00021     string getName();
00022     int getID();
00023     bool getVal();
00024 
00025     OUTPUT();
00026     OUTPUT(int iID);
00027     OUTPUT(gate *ing, int iID);
00028     void setInputs(gate *ing, gate *n);
00029 
00036     bool calculate();
00037     ~OUTPUT();
00038 
00039     void write(std::ostream& os = std::cout);
00040 };
00041 
00042 
00043 #endif
```

6.19 XOR.h File Reference

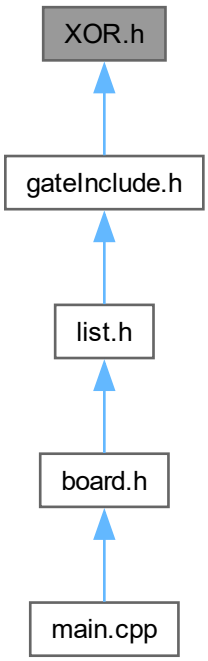
Implementation of the XOR gate, 2 inputs.

```
#include <iostream>
#include <string>
#include ¨gate.h¨
```

Include dependency graph for XOR.h:



This graph shows which files directly or indirectly include this file:



Classes

- class XOR

6.19.1 Detailed Description

Implementation of the XOR gate, 2 inputs.

6.20 XOR.h

Go to the documentation of this file.

```
00001
00006 #ifndef XORG
00007 #define XORG
00008
00009 #include <iostream>
00010 #include <string>
00011 #include "gate.h"
00012
00013 using std::string;
00014 using std::cout;
00015 using std::endl;
00016
00017 class XOR : public gate
00018 {
00019     string name = "XOR";
00020 public:
```

```
00021     string getName();
00022     int getID();
00023     bool getVal();
00024 
00025     XOR();
00026     XOR(int iID);
00027     XOR(gate *i1, gate *i2, int iID);
00028     void setInputs(gate *i1, gate *i2);
00029 
00036     bool calculate();
00037     ~XOR();
00038 
00039     void write(std::ostream& os = std::cout);
00040 };
00041 
00042 
00043 #endif
```

Index